



Composant Serveur API REST pour WinDev®

Léger.

Rapide.

Gratuit & Open Source.

Sans serveur IIS/Apache/...

Sans licence Serveur WebDev®.

Pas de SAAS, pas d'abonnement.

Windows 32/64 & Linux

Présentation

NEW : LightREST V3

La Documentation



Téléchargements

Un coup de main ?

Les **HOOKS** (ou *intercepteurs d'évènements*) permettent d'exécuter du code automatiquement à des moments clés du cycle de vie d'une requête LightREST.

Ils servent à gérer proprement tous les traitements **transverses** (non métier), sans polluer le code des routes : sécurité, logs, métriques, traçage, normalisation des réponses, gestion centralisée des erreurs, etc.

Un handler de Hook reçoit en paramètre une structure EventInfo qui contient :

- Le type de l'évènement intercepté
- l'objet lrRequest de la requête en cours (dont il peut renseigner le membre CustomData à l'attention de handlers qui seront exécutés ultérieurement)
- l'objet lrResponse (à compléter éventuellement)
- un éventuel message (pour les événements EVE_INFO, EVE_WARNING, EVE_ERROR et EVE_PANIC)

– le Tag de la route en cours d'exécution

Un handler de Hook renvoie une valeur de type EventReturn :

- EVE_OK : le traitement de la route continue normalement
- EVE_ABORT : le traitement est interrompu. Le contenu de la réponse (headers+body) est envoyé au client (avec une probable erreur HTTP)
- EVE_REPLACE : utilisé pour un événement EVE_BEFORE_HANDLER il permet de court-circuiter l'exécution du handler par défaut de la route en cours

Le projet WinDev LightRestHooksDemo (présent dans la distribution à partir de la v3.2) illustre les fondamentaux des Hooks.

Intérêt

Un hook est une fonction appelée par LightREST à un instant précis du traitement d'une requête.

Il permet notamment de :

- Exécuter un **traitement alternatif** qui remplacera celui de la ROUTE appelée
- **Filter / compléter / remplacer / modifier** le contenu d'une réponse REST
- **Bloquer** l'utilisation d'une API si certains critères ne sont pas remplis (ex: géographiques), ou interdire l'exécution d'une ROUTE (droits)
- **Injecter des données** dans l'objet IrRequest reçu par un handler REST (membre CustomData)
- Centraliser la **journalisation** (logs structurés, durée, statut HTTP...)
- Implémenter l'**authentification / autorisation** de manière

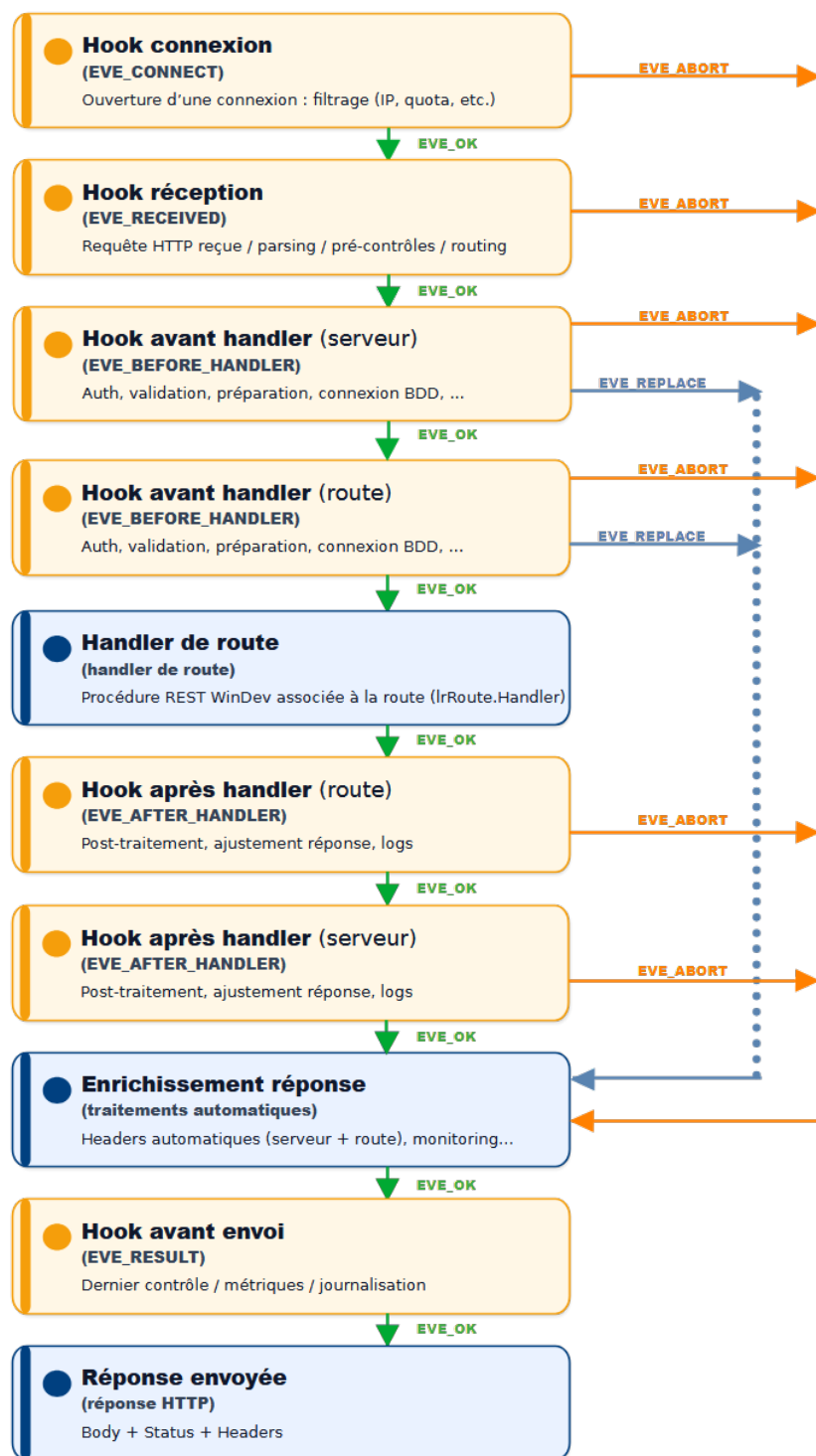
uniforme

- Ajouter du **traçage** (Correlation ID, audit, conformité)
 - Collecter des **métriques de performance**
 - Appliquer des règles globales (CORS, headers de sécurité, cache-control...)
 - Normaliser les **réponses** et les **erreurs**
 - **Éviter la duplication** de code dans chaque route
-

Avantages clés

- **Cohérence** : mêmes règles appliquées partout
 - **Lisibilité** : les routes restent focalisées sur le métier
 - **Maintenance** : un changement transverse se fait en un seul endroit
 - **Robustesse** : moins d'oublis, moins de divergences de comportement
-

Cycle de vie d'une requête et position des hooks



Le traitement typique d'une requête REST suit ce schéma :

1. Ouverture d'une connexion (EVE_CONNECT)
2. Réception de la requête REST (EVE_RECEIVED)
3. Exécution du hook *avant handler* (EVE_BEFORE_HANDLER)

4. Exécution du handler de route
5. Exécution du hook *après handler* (EVE_BEFORE_HANDLER)
6. Ajout des éventuels headers automatiques (niveau serveur et route)
7. Exécution du hook *avant envoi de la réponse* (EVE_RESULT)
8. Envoi du résultat au client REST

En cas d'erreur (erreur applicative, panic, timeout...), des hooks spécifiques peuvent être exécutés afin de produire une réponse cohérente et effectuer les nettoyages nécessaires.

⚠ Certains hooks peuvent être exécutés même si le client a déjà coupé la connexion (timeout d'écriture, socket fermé).

Principe

- Un ou plusieurs hooks sont **enregistrés** sur le serveur ou sur une route.
- Chaque hook est associé à une **phase d'exécution** (EVE_CONNECT, EVE_BEFORE_HANDLER, EVE_ERROR, ...) .
- Un hook reçoit le **contexte de requête** et peut :
 - Lire ou modifier la requête et la réponse
 - Ajouter des informations contextuelles
 - Stopper le traitement
 - Déclencher une erreur contrôlée

Règles importantes

- Un hook doit être **rapide** (éviter toute I/O bloquante).
- Il doit être **sûr en concurrence** (aucun état global mutable non protégé).

- Il ne doit jamais exposer de **données sensibles** dans les logs.
- Les décisions globales (auth, CORS, headers communs) doivent être traitées ici, pas dans les routes.

Evénements du mécanisme de hooks

Les Hooks peuvent être déclarés niveau Serveur et/ou Route selon leur type. Un même événement « Hooké » sur les 2 niveaux sera d'abord exécuté sur l'objet Serveur et ensuite sur l'objet Route.

Plusieurs événements peuvent être attachés à un même Hook (par exemple un même handler pourra gérer EVE_ERROR, EVE_WARNING et EVE_PANIC).

EVE_INFO	Hook serveur. Information sur l'exécution
EVE_STARTING	Hook serveur. En cours de démarrage
EVE_STARTED	Hook serveur. Démarré
EVE_STOPPING	Hook serveur. En cours d'arrêt
EVE_STOPPED	Hook serveur. Arrêté
EVE_CONNECT	Hook serveur. Nouvelle connexion (permet par exemple de filtrer les connexions par adresse IP)
	Hook serveur et route. Réception

EVE_RECEIVED	d'une requête REST
EVE_AUTH_FAILED	Hook serveur et route. Authentification échouée
EVE_BEFORE_HANDLER	Hook serveur et route. Avant exécution de la route
EVE_AFTER_HANDLER	Hook serveur et route. Après exécution de la route
EVE_RESULT	Hook serveur et route. Avant envoi du résultat (étape ultime pour vérifier, compléter, chiffrer, signer, filtrer, limiter, journaliser, ...) la réponse avant envoi au client.
EVE_ERROR	Hook serveur. Erreur lors de l'exécution
EVE_WARNING	Hook serveur. Warning lors de l'exécution
EVE_PANIC	Hook serveur. Erreur grave lors de l'exécution

Exemple

- Authentification centralisée via un hook serveur
- Vérification d'un token dans le header *Authorization*
- Retour **401 Unauthorized** avec une réponse JSON standardisée

- Aucune route n'a besoin de gérer l'authentification

Copier

```
//
=====

=====
// Hook événementiel : gestion de
l'authentification
//
=====

=====
PROCÉDURE ServerEventHook(stInfo
lrHook.lrEventInfo) :
lrHook.lrEventReturn

    SI
stInfo.Request.GetHeaderValue("ID")=""
ALORS

        stInfo.Response:Status      =
lrResponse::StatusUnauthorized
        stInfo.Response:ContentType =
lrResponse::ContentTXT
        stInfo.Response:Body        =
"Token ID manquant"

        // Stoppe le pipeline : la route
ne sera pas exécutée
        RENVOYER lrHook::EVE_ABORT
    FIN

    RENVOYER lrHook::EVE_OK
```

FIN

© CODE LINE 2018-2026

